Improving and Enhancing SACK

Anthony Sabatini
Formerly CTO Telerate Inc.

The problem – packet loss

- Interference with Wifi
- Interference with Mobile Phones
- Rate caps on mobile phones
- Traffic Shapers
- Content monitoring that can not keep up
- Overloaded links

The list is endless and getting worse

The Issue – TCP Error recovery

- TCP is a simple protocol to which a tremendous amount of effort was applied to improve error recovery outside the protocol.
- TCP until recently was sufficient because error rates constantly went down. They are now rising, especially error bursts.
- Selective Acknowledgement (SACK) is the first major error recovery improvement in the protocol.

Current SACK Limitations

- Recovery is timer based and timers must be set to worst case
- Insufficient repetition of SACK information
- Resegmented segments are entirely retrans
- No long (RTT) SACK retransmission strategy
- No recovery of lost retransmits
- Out of order segments cause spurious retrans
- Last transmitted segment(s) loss unrecovered

Suggested Improvements to SACK

- A better way to fill SACK block slots in ACK
- Transmission of link idle ACK so last changed SACK block gets second transmission.
- A preemptive retransmission of all SACK blocks if oldest SACK block goes unfilled for 1.25*RTT.

Enhanced SACK - Event Driven

- Send Token is added to each message, its return allows sender to know what the receiver should have seen and what has already been retransmitted
- Send Token automatically rebuilds optimal retransmit list when it is returned
- Round trip times easily calculated without other constructs

Compatibility

- Improvements are transparent to all current SACK implementations
- Enhancements can be ignored on links that do not support them, reverts back to timers

Thought Points

- Enhanced SACK is very robust in highly congested environments
- Event driven so it is as fast as link can support
- Enhanced SACK does not increase bandwidth requirements significantly which is more than made up by duplicate packet elimination

Typical Exchange

```
Thus (p = packet, t = token) -
Transmit - Network   -    Recvieve
P1T0       -->
P2T1       -->
P3T2       -->
P4T3       -->
(Waits for work)
           --X                (first packet lost)
           P2T1               -->
           <--                SACK T1P2
           --X                (third packet lost)
           P4T3               -->
           <--                SACK T3P2P4
                              (Waits for work)
<--        SACK T1P2
P1T4       -->
<--        SACK T3P2P4
P3T5       -->
(It knows P1 is "inflight")
           P1T4               -->
           <--                ACK 2 SACK T4P4
           P3T5               -->
           <--                ACK 4 SACK T5
<--        ACK 2 SACK T4P2
(Moves new floor to 2)
(It knows P3 "inflight")
<--        ACK 4 SACK T5
(Moves new floor to 4)
```